Optimal lower bounds for universal relation, and for samplers and finding duplicates in streams

Michael Kapralov[*], Jelani Nelson[†], Jakub Pachocki[‡],
Zhengyu Wang[§], David P. Woodruff[¶] and Mobin Yahyazadeh [‖]

[*]EPFL. michael.kapralov@epfl.ch.
[†]Harvard University. minilek@seas.harvard.edu.
[‡] OpenAI. jakub@openai.com.
[§]Harvard University. zhengyuwang@g.harvard.edu.
[¶]Carnegie Mellon University. dwoodruf@cs.cmu.edu.
[‖] Sharif University of Technology. mn.yahyazadeh@gmail.com

Abstract—In the communication problem UR (universal relation) [25], Alice and Bob respectively receive $x, y \in \{0,1\}^n$ with the promise that $x \neq y$. The last player to receive a message must output an index i such that $x_i \neq y_i$. We prove that the randomized one-way communication complexity of this problem in the public coin model is exactly $\Theta(\min\{n, \log(1/\delta) \log^2(\frac{n}{\log(1/\delta)})\})$ for failure probability δ. Our lower bound holds even if promised $support(y) \subset support(x)$. As a corollary, we obtain optimal lower bounds for ℓ_p-sampling in strict turnstile streams for $0 \leq p < 2$, as well as for the problem of finding duplicates in a stream. Our lower bounds do not need to use large weights, and hold even if promised $x \in \{0,1\}^n$ at all points in the stream.

We give two different proofs of our main result. The first proof demonstrates that any algorithm $\mathcal{A}$ solving sampling problems in turnstile streams in low memory can be used to encode subsets of $[n]$ of certain sizes into a number of bits below the information theoretic minimum. Our encoder makes adaptive queries to $\mathcal{A}$ throughout its execution, but done carefully so as to not violate correctness. This is accomplished by injecting random noise into the encoder's interactions with $\mathcal{A}$, which is loosely motivated by techniques in differential privacy. Our correctness analysis involves understanding the ability of $\mathcal{A}$ to correctly answer adaptive queries which have positive but bounded mutual information with $\mathcal{A}$'s internal randomness, and may be of independent interest in the newly emerging area of adaptive data analysis with a theoretical computer science lens. Our second proof is via a novel randomized reduction from Augmented Indexing [30] which needs to interact with $\mathcal{A}$ adaptively. To handle the adaptivity we identify certain likely interaction patterns and union bound over them to guarantee correct interaction on all of them. To guarantee correctness, it is important that the interaction hides some of its randomness from $\mathcal{A}$ in the reduction.

Keywords—streaming, lower bounds, ℓ_p-sampling

I. Introduction

In turnstile ℓ_0-sampling, a vector $z \in \mathbb{R}^n$ starts as the zero vector and receives coordinate-wise updates of the form "$z_i \leftarrow z_i + \Delta$" for $\Delta \in \{-M, -M + 1, \ldots, M\}$. During a query, one must return a uniformly random element from $support(z) = \{i : z_i \neq 0\}$. The problem was first defined in [15], where a data structure (or "sketch") solving it was used to estimate the Euclidean minimum spanning tree, and to provide ε-approximations of a point set in a geometric space. Sketches for ℓ_0-sampling were also used to solve various dynamic graph streaming problems in [1] and since then have been crucially used in almost all known dynamic graph streaming algorithms[1], such as for: connectivity, k-connectivity, bipartiteness, and minimum spanning tree [1], subgraph counting, minimum cut, and cut-sparsifier and spanner computation [2], spectral sparsifiers [3], maximal matching [10], maximum matching [1], [8], [27], [6], [9], [5], vertex cover [10], [9], hitting set, b-matching, disjoint paths, k-colorable subgraph, and several other problems [9], densest subgraph [7], [29], [13], vertex and hyperedge connectivity [18], and graph degeneracy [14]. For an introduction to the power of ℓ_0-sketches in designing dynamic graph stream algorithms, see [28, Section 3]. Such sketches have also been used outside streaming, such as in distributed algorithms [19], [34] and data structures for dynamic connectivity [24], [37], [16].

Given the rising importance of ℓ_0-sampling in algorithm design, a clear task is to understand the exact

---

[1]The spectral sparsification algorithm of [22] is a notable exception.

IEEE
computer
society

reference	problem	distribution	$k > 1$?	$\delta = o(1)$?
[15]	Euclidean minimum spanning tree	ℓ_0	yes	
[1]	connectivity[2]	any		yes
[1]	k-connectivity[2]	any		yes
[1]	bipartiteness[2]	any		yes
[1]	minimum spanning tree	any		yes
[2]	subgraph counting	ℓ_0	yes	
[2]	minimum cut	any		yes
[2]	cut sparsifiers	any		yes
[2]	spanners	any	yes	yes
[2]	spectral sparsifiers	any		yes
[10]	maximal matching	ℓ_0	yes	yes
[8]	maximum matching (unweighted)	ℓ_0	yes	
	maximum matching (weighted)	ℓ_0	yes	yes
[27]	maximum matching	any	yes	yes
[6]	maximum matching	ℓ_0		yes
[5]	maximum matching	ℓ_0		yes
[9]	maximum matching	ℓ_0		yes
	vertex cover			
	hitting set			
	b-matching			
	disjoint paths			
	k-colorable subgraph			
[7]	densest subgraph	ℓ_0		yes
[29]	densest subgraph	ℓ_0	yes	yes
[13]	densest subgraph	ℓ_0	yes	
[18]	vertex connectivity	any		yes
	hyperedge connectivity			
[14]	graph degeneracy	ℓ_0	yes	

Fig. 1. Guarantees needed by various works using samplers as subroutines. The last two columns indicate whether the work needs to use a sampler that returns k samples at a time when queried for some $k > 1$, or for some subconstant failure probability δ even to achieve failure probability $1/3$ in the main application. The "distribution" column indicates the output distribution needed from the sampler for the application ("any" means a support-finding subroutine is sufficient, i.e., it suffices for a query to return any index i for which $z_i \neq 0$).

complexity of this problem. The work [20] gave an $\Omega(\log^2 n)$-bit space lower bound for data structures which solve even the case $M = 1$ and which fail with constant probability δ (meaning query responses are δ-close to the uniform in $support(z)$). They also gave an upper bound for $M \leq \mathrm{poly}(n)$ with failure probability δ, which in fact gave $\min\{\|z\|_0, \Theta(\log(1/\delta))\}$ uniform samples from the support of z, using space $O(\log^2 n \log(1/\delta))$ (here $\|z\|_0$ denotes $|support(z)|$). Thus we say their data structure actually solves the harder problem of ℓ_0-sampling$_k$ for $k = \Theta(\log(1/\delta))$ with failure probability δ, where in ℓ_0-sampling$_k$ the goal is to recover $\min\{\|z\|_0, k\}$ uniformly random elements, without replacement, from $support(z)$. The upper and lower bounds in [20] thus match up to a constant factor for $k = 1$ and δ a constant. We note though in many settings, even if the final application desires constant failure probability, ℓ_0-sampling$_k$ with either failure probability $o(1)$ or $k > 1$ (or both) is needed as a subroutine (see Figure 1).

Universal relation. The work of [20] obtains its lower bound for ℓ_0-sampling (and some other problems) via reductions from *universal relation* (**UR**). This problem was defined in [25] and arose in connection with the work [26] on circuit depth lower bounds. In this problem, Alice and Bob are given $x, y \in \{0, 1\}^n$ with the promise $x \neq y$. The players must then agree on any index i with $x_i \neq y_i$ (in the variant we study, we are satisfied if only one of the players determines such an i). The deterministic communication complexity of **UR** is nearly completely understood, with upper and lower bounds that match up to an additive 3 bits, even if one imposes an upper bound on the number of rounds of communication [35]. Henceforth we also consider a generalized problem **UR**$_k$, where the output must be $\min\{k, \|x - y\|_0\}$ distinct indices on which x, y differ. We also use **UR**$^\subset$, **UR**$_k^\subset$ to denote the variants when promised $support(y) \subset support(x)$, and also Bob knows $\|x\|_0$. Clearly **UR**, **UR**$_k$ can only be harder

than $\mathbf{UR}^{\subseteq}, \mathbf{UR}_k^{\subseteq}$, respectively.

More than twenty years after its initial introduction in connection with circuit depth lower bounds, Jowhari et al. in [20] demonstrated the relevance of $\mathbf{UR}$ in the randomized one-way communication model for obtaining space lower bounds for certain streaming problems, such as various sampling problems and finding duplicates in streams. In the one-way version, Bob simply needs to find such an index i after a single message from Alice, and we only charge Alice's single message's length as the communication cost. If $\mathbf{R}_\delta^{\rightarrow,pub}(f)$ denotes the randomized one-way communication complexity of f in the public coin model with failure probability δ, [20] showed that the space complexity of $\mathsf{FindDuplicate}(n)$ with failure probability δ is at least $\mathbf{R}_{\frac{7}{8}+\frac{\delta}{8}}^{\rightarrow,pub}(\mathbf{UR})$. In $\mathsf{FindDuplicate}(n)$, one is given a length-$(n+1)$ stream of integers in $[n]$, and the algorithm must output some element $i \in [n]$ which appeared at least twice in the stream (note that at least one such element must exist, by the pigeonhole principle). The work [20] then showed a reduction demonstrating that any solution to ℓ_0-sampling with failure probability δ in turnstile streams immediately implies a solution to $\mathsf{FindDuplicate}(n)$ with failure probability at most $(1+\delta)/2$ in the same space (and thus the space must be at least $\mathbf{R}_{\frac{15}{16}+\frac{\delta}{16}}^{\rightarrow,pub}(\mathbf{UR})$). The same result is shown for ℓ_p-sampling for any $p > 0$, in which the output index should equal i with probability $|x_i|^p/(\sum_j |x_j|^p)$, and a similar result is shown even if the distribution on i only has to be close to this ℓ_p-distribution in variational distance (namely, the distance should be bounded away from 1). It is then shown in [20] that $\mathbf{R}_\delta^{\rightarrow,pub}(\mathbf{UR}) = \Omega(\log^2 n)$ for any δ bounded away from 1. The approach used though does not provide an improved lower bound for $\delta \to 0$.

Seemingly unnoticed in [20], we first point out here that the lower bound proof for $\mathbf{UR}$ in that work actually proves the same lower bound for the promise problem $\mathbf{UR}^{\subseteq}$. This observation has several advantages. First, it makes the reductions to the streaming problems trivial (they were already quite simple when reducing from $\mathbf{UR}$, but now they are even simpler). Second, a simple reduction from $\mathbf{UR}^{\subseteq}$ to sampling problems provides space lower bounds even in the strict turnstile model,

and even for the simpler *support-finding* streaming problem for which when queried is allowed to return *any* element of $support(z)$, without any requirement on the distribution of the index output. Both of these differences are important for the meaningfulness of the lower bound. This is because in dynamic graph streaming applications, typically z is indexed by $\binom{n}{2}$ for some graph on n vertices, and z_e is the number of copies of edge e in some underlying multigraph. Edges then are not deleted unless they had previously been inserted, thus only requiring correctness for strict turnstile streams. Also, for every single application mentioned in the first paragraph of Section I (except for the two applications in [15]), the known algorithmic solutions which we cited as using ℓ_0-sampling as a subroutine actually only need a subroutine for the easier support-finding problem. Finally, third and most relevant to our current work's main focus, the straightforward reductions from $\mathbf{UR}^{\subseteq}$ to the streaming problems we are considering here do not suffer any increase in failure probability, allowing us to transfer lower bounds on $\mathbf{R}_\delta^{\rightarrow,pub}(\mathbf{UR}^{\subseteq})$ for small δ to lower bounds on various streaming problems for small δ. The work [20] could not provide lower bounds for the streaming problems considered there in terms of δ for small δ.

We now show simple reductions from $\mathbf{UR}^{\subseteq}$ to $\mathsf{FindDuplicate}(n)$ and from $\mathbf{UR}_k^{\subseteq}$ to support-finding$_k$. In support-finding$_k$ we must report $\min\{k, \|z\|_0\}$ elements in $support(z)$. In the claims below, δ is the failure probability for the considered streaming problem.

Claim 1. *Any one-pass streaming algorithm for* $\mathsf{FindDuplicate}(n)$ *must use* $\mathbf{R}_\delta^{\rightarrow,pub}(\mathbf{UR}^{\subseteq})$ *space.*

Proof: We reduce from $\mathbf{UR}^{\subseteq}$. Suppose there were a space-S algorithm $\mathcal{A}$ for $\mathsf{FindDuplicate}(n)$. Alice creates a stream consisting of all elements of $support(x)$ and runs $\mathcal{A}$ on those elements, then sends the memory contents of $\mathcal{A}$ to Bob. Bob then continues running $\mathcal{A}$ on $n+1-\|x\|_0$ arbitrarily chosen elements of $[n] \setminus support(y)$. Then there must be a duplicate in the resulting concatenated stream, i satisfies $x_i \neq y_i$ iff i is a duplicate. ∎

The two claims below have similarly simple proofs, which are provided in the full version.

Claim 2. *Any one-pass streaming algorithm for* support-finding$_k$ *in the strict turnstile model must use* $\mathbf{R}_\delta^{\rightarrow,pub}(\mathbf{UR}_k^{\subseteq})$ *bits of space, even if promised that* $z \in \{0,1\}^n$ *at all points in the stream.*

Claim 3. *Any one-pass streaming algorithm for* ℓ_p-*sampling for any* $p \geq 0$ *in the strict turnstile model must use* $\mathbf{R}_\delta^{\rightarrow,pub}(\mathbf{UR}_k^{\subseteq})$ *bits of space, even if promised* $z \in \{0,1\}^n$ *at all points in the stream.*

---

[2][1] describes these algorithms as only needing δ a constant, but for a different definition of support-finding: when the data structure fails, it should output Fail instead of behaving arbitrarily. They then cite [20] as providing the sampler they use, but unfortunately [20] does not solve this variant of this problem. This issue can be avoided by using [20] with $\delta < 1/\mathrm{poly}(n)$ so that with high probability no failures occur throughout their algorithm, very slightly worsening their final space bound in the main application by a $\lg n$ factor. Alternatively, one could use another variant of sampling which we define in the full version [23] and which can be solved via a minor modification of [20], which allows retaining the bounds of [1] for connectivity, k-connectivity, and bipartiteness testing.

The reductions above thus raise the question: what is the asymptotic behavior of $\mathbf{R}_\delta^{\rightarrow,pub}(\mathbf{UR}_k^\subset)$?

Our main contribution: We prove for any δ bounded away from 1 and $k \in [n]$, $\mathbf{R}_\delta^{\rightarrow,pub}(\mathbf{UR}_k^\subset) = \Theta(\min\{n, t\log^2(n/t)\})$ where $t = \max\{k, \log(1/\delta)\}$. Given known upper bounds in [20], our lower bounds are optimal for $\mathsf{FindDuplicate}(n)$, support-finding, and ℓ_p-sampling for any $0 \le p < 2$ for nearly the full range of n, δ (namely, for $\delta > 2^{-n^{.99}}$). Also given an upper bound of [20], our lower bound is optimal for ℓ_0-sampling$_k$ for nearly the full range of parameters n, k, δ (namely, for $t < n^{.99}$). Previously no lower bounds were known in terms of δ (or k). Our main theorem:

Theorem 1. *For any δ bounded away from 1 and $1 \le k \le n$, $\mathbf{R}_\delta^{\rightarrow,pub}(\mathbf{UR}_k^\subset) = \Theta(\min\{n, t\log^2(n/t)\})$.*

We give two different proofs of Theorem 1 (in Sections III and IV). Our upper bound is also new, though follows by minor modifications of the upper bound in [20] and thus we describe it in the full version [23]. The previous upper bound was $O(\min\{n, t\log^2 n\})$. We also mention here that it is known that the upper bound for both $\mathbf{UR}_k$ and ℓ_0-sampling$_k$ in two rounds (respectively, two passes) is only $O(t\log n)$ [20]. Thus, one cannot hope to extend our new lower bound to two or more passes, since it simply is not true.

A. Related work

The question of whether ℓ_0-sampling is possible in low memory in turnstile streams was first asked in [12], [15]. The work [15] applied ℓ_0-sampling as a subroutine in approximating the cost of the Euclidean minimum spanning tree of a subset S of a discrete geometric space subject to insertions and deletions. The algorithm given there used space $O(\log^3 n)$ bits to achieve failure probability $1/\mathrm{poly}(n)$ (though it is likely that the space could be improved to $O(\log^2 n \log\log n)$ with a worse failure probability, by replacing a subroutine used there with a more recent ℓ_0-estimation algorithm of [21]). As mentioned, the currently best known upper bound solves ℓ_0-sampling$_k$ using $O(t\log^2 n)$ bits [20], which Theorem 1 shows is tight.

For ℓ_p-sampling, conditioned on not failing, the data structure should output i with probability $(1 \pm \varepsilon)|x_i|^p/\|x\|_p^p$. The first work to realize its importance came even earlier than for ℓ_0-sampling: [11] showed that an ℓ_2-sampler using small memory would lead to a nearly space-optimal streaming algorithm for multiplicatively estimating $\|x\|_3$ in the turnstile model, but did not know how to implement such a data structure. The first implementation was given in [31], achieving space $\mathrm{poly}(\varepsilon^{-1}\log n)$ with $\delta = 1/\mathrm{poly}(n)$. . For $1 \le p \le 2$ the space was improved to $O(\varepsilon^{-p}\log^3 n)$ bits for constant δ [4]. In [20] this bound was improved

to $O(\varepsilon^{-\max\{1,p\}}\log(1/\delta)\log^2 n)$ bits for failure probability δ when $0 < p < 2$ and $p \ne 1$. For $p = 1$ the space bound achieved by [20] was a $\log(1/\varepsilon)$ factor worse: $O(\varepsilon^{-1}\log(1/\varepsilon)\log(1/\delta)\log^2 n)$ bits.

For finding a duplicate item in a stream, the question of whether a space-efficient randomized algorithm exists was asked in [32], [36]. The question was positively resolved in [17], which gave an $O(\log^3 n)$-space algorithm with constant failure probability. An improved algorithm was given in [20], using $O(\log(1/\delta)\log^2 n)$ bits of space for failure probability δ.

II. OVERVIEW OF TECHNIQUES

We now describe our two proofs of Theorem 1. For the upper bound, [20] achieved $O(t\log^2 n)$, but in the full version [23] we show that slight modifications to their approach yield $O(\min\{n, t\log^2(n/t)\})$. Our main contribution is in proving an improved lower bound. Assume $t < cn$ for some sufficiently small constant c (since otherwise we already obtain an $\Omega(n)$ lower bound). In both our lower bound proofs in this regime, the proof is split into two parts: we show $\mathbf{R}_\delta^{\rightarrow,pub}(\mathbf{UR}^\subset) = \Omega(\log\frac{1}{\delta}\log^2\frac{n}{\log\frac{1}{\delta}})$ and $\mathbf{R}_{.99}^{\rightarrow,pub}(\mathbf{UR}_k^\subset) = \Omega(k\log^2\frac{n}{k})$ separately. We give an overview the former here, which is the more technically challenging half. Our two proofs of the latter are in the full version.

A. Lower bound proof via encoding subsets and an adaptivity lemma

Our first proof of the lower bound on $\mathbf{R}_\delta^{\rightarrow,pub}(\mathbf{UR}^\subset)$ is via an encoding argument. Fix m. A randomized encoder is given a set $S \subset [n]$ with $|S| = m$ and must output an encoding $\mathsf{ENC}(S)$. A decoder, who shares public randomness with the encoder, must then be able to recover S given only $\mathsf{ENC}(S)$. We consider such schemes in which the decoder must succeed with probability 1, and the encoding length is a random variable. Any such encoding must use $\Omega(\log\binom{n}{m}) = \Omega(m\log\frac{n}{m})$ bits in expectation for some S.

There is a natural, but sub-optimal approach to using a public-coin one-way protocol $\mathcal{P}$ for $\mathbf{UR}^\subset$ to devise such an encoding/decoding scheme. The encoder pretends to be Alice with input x being the indicator set of S, then lets $\mathsf{ENC}(S)$ be the message M that Alice would have sent to Bob. The decoder attempts to recover S by iteratively pretending to be Bob m times, initially pretending to have input $y = 0 \in \{0,1\}^n$, then iteratively adding elements found in S to y's support. Henceforth let $\mathbf{1}_T \in \{0,1\}^n$ denote the indicator vector of a set $T \subset [n]$.

Algorithm 1 Simple Decoder.

1: **procedure** DEC(M)
2:     $T \leftarrow \emptyset$
3:     **for** $r = 1, \ldots, m$ **do**
4:         Let i be Bob's output upon receiving message M from Alice when Bob's input is $\mathbf{1}_T$
5:             $T \leftarrow T \cup \{i\}$
6:     **end for**
7:     **return** T
8: **end procedure**

One might hope to say that if the original failure probability were $\delta < 1/m$, then by a union bound, with constant probability every iteration succeeds in finding a new element of S (or one could even first apply some error-correction to x so that the decoder could recover S even if only a constant fraction of iterations succeeded). The problem with such thinking though is that this decoder chooses y's adaptively! To be specific, $\mathcal{P}$ being a correct protocol means

$$\forall x, y \in \{0,1\}^n, \; \mathbb{P}_s(\mathcal{P} \text{ is correct on inputs } x, y) \geq 1-\delta, \quad (1)$$

where s is the public random string that both Alice and Bob have access to. The issue is that even in the second iteration (when $r = 2$), Bob's "input" $\mathbf{1}_T$ *depends on s*, since T depends on the outcome of the first iteration! Thus the guarantee of (1) does not apply.

One way around the above issue is to realize that as long as every iteration succeeds, T is always a subset of S. Thus it suffices for the following event $\mathcal{E}$ to occur: $\forall T \subset S$, $\mathcal{P}$ is correct on inputs $\mathbf{1}_S, \mathbf{1}_T$. Then $\mathbb{P}_s(\neg \mathcal{E}) \leq 2^m \delta$ by a union bound, which is at most $1/2$ for $m = \lfloor \log_2(1/\delta) \rfloor - 1$. We have thus just shown that $\mathbf{R}_\delta^{\rightarrow, pub}(\mathbf{UR}^\subset) = \Omega(\min\{n, \log\binom{n}{m}\}) = \Omega(\min\{n, \log\frac{1}{\delta} \log \frac{n}{\log(1/\delta)}\})$.

Our improvement is as follows. Our new decoder again iteratively tries to recover elements of S as before. We will give up though on having m iterations and hoping for all (or even most) of them to succeed. Instead, we will only have $R = \Theta(\log \frac{1}{\delta} \log \frac{n}{\log \frac{1}{\delta}})$ iterations, and our aim is for the decoder to succeed in finding a new element in S for at least a constant fraction of these R iterations. Simplifying things for a moment, let us pretend for now that all R iterations do succeed in finding a new element. $\mathsf{ENC}(S)$ will then be Alice's message M, together with the set $B \subset S$ of size $m - R$ not recovered during the R rounds, explicitly written using $\lceil \log\binom{n}{|B|} \rceil$ bits. If the decoder can then recover these R remaining elements, this then implies the decoder has recovered S, and thus we must have $|M| = \Omega(\log\binom{n}{m} - \log\binom{n}{|B|}) = \Omega(R \log \frac{n}{m})$. The decoder proceeds as follows. Just as before, initially the decoder starts with $T = \emptyset$ and lets i be the output of

Bob on $\mathbf{1}_T$ and adds it to T. Then in iteration r, before proceeding to the next iteration, the decoder randomly picks some elements from B and adds them into T, so that the number of elements left to be uncovered is some fixed number n_r. These extra elements being added to T should be viewed as "random noise" to mask information about the random string s used by $\mathcal{P}$, an idea very loosely inspired by ideas in differential privacy. For intuition, as an example suppose the iteration $r = 1$ succeeds in finding some $i \in S$. If the decoder were then to add i to T, as well as $\approx m/2$ random elements from B to T, then the resulting T reveals only ≈ 1 bit of information about i (and hence about s). This is as opposed to the $\log m$ bits T could have revealed if the masking were not performed. Thus the next query in round $r = 2$, although correlated with s, has very weak correlation after masking and we thus might hope for it to succeed. This intuition is captured in the following lemma, which we prove in the full version:

Lemma 1. *Consider* $f\colon \{0,1\}^b \times \{0,1\}^q \to \{0,1\}$ *and* $X \in \{0,1\}^b$ *uniformly random. If* $\forall y \in \{0,1\}^q$, $\mathbb{P}(f(X,y) = 1) \leq \delta$ *where* $0 < \delta < 1$, *then for any random variable Y supported on $\{0,1\}^q$,*

$$\mathbb{P}(f(X,Y) = 1) \leq \frac{I(X;Y) + H_2(\delta)}{\log \frac{1}{\delta}}, \quad (2)$$

where $I(X;Y)$ is the mutual information between X and Y, and H_2 is the binary entropy function.

Fix some $x \in \{0,1\}^n$. One should imagine here that $f(X,y)$ is 1 iff $\mathcal{P}$ fails when Alice has input x and Bob has input y in a $\mathbf{UR}^\subset$ instance, and the public random string is $X = s$. Then the lemma states that if $y = Y$ is not arbitrary, but rather random (and correlated with X), then the failure probability of the protocol is still bounded as long as the mutual information between X and Y is bounded. It is also not hard to see that this lemma is sharp up to small additive terms. Consider the case $x, y \in [n]$, and $f(x,y) = 1$ iff $x = y$. Then if X is uniform, for all y we have $\mathbb{P}(f(X,y) = 1) = 1/n$. Now consider the case where Y is random and equal to X with probability $t/\log n$ and is uniform in $[n]$ with probability $1 - t/\log n$. Then in expectation Y reveals t bits of X, so that $I(X;Y) = t$. It is also not hard to see that $\mathbb{P}(f(X,Y) = 1) \approx t/\log n + 1/n$.

In light of the strategy stated so far and Lemma 1, the path forward is clear: at each iteration r, we should add enough random masking elements to T to keep the mutual information between T and all previously added elements below, say, $\frac{1}{2} \log \frac{1}{\delta}$. Then we expect a constant fraction of iterations to succeed. The encoder knows which iterations do not succeed since it shares public randomness with the decoder (and can thus simulate it), so it can simply tell the decoder which rounds are the failed ones, then explicitly include in M correct

new elements of S for the decoder to use in the place of Bob's wrong output in those rounds. A calculation shows that if one adds a $(1 - 1/K) \approx 2^{-1/K}$ fraction of the remaining items in S to T after drawing one more support element from Bob, the mutual information between the next query to Bob and the randomness used by $\mathcal{P}$ will be $O(K)$ (see Lemma 5). Thus we do this for K a sufficiently small constant times $\log \frac{1}{\delta}$. We will then have $n_r \approx (1 - 1/K)^r m$. Note that we cannot continue in this way once $n_r < K$ (since the number of "random noise" elements we inject should at least be one). Thus we are forced to stop after $R = \Theta(K \log(m/K)) = \Theta(\log \frac{1}{\delta} \log \frac{n}{\log \frac{1}{\delta}})$ iterations. We then set $m = \sqrt{n \log(1/\delta)}$, so that $\mathbf{R}_\delta^{\rightarrow, pub}(\mathbf{UR}^\subset) = \Omega(|R| \log \frac{n}{m}) = \Omega(\min\{n, \log \frac{1}{\delta} \log^2 \frac{n}{\log \frac{1}{\delta}}\})$ as desired.

The argument for lower bounding $\mathbf{R}_\delta^{\rightarrow, pub}(\mathbf{UR}_k^\subset)$ is a bit simpler, and in particular does not rely on Lemma 1. It can be found in the full version.

As mentioned above, our lower bounds use protocols for $\mathbf{UR}^\subset$ and $\mathbf{UR}_k^\subset$ to establish protocols for encoding subsets of some fixed size m of $[n]$. These encoders always consist of some message M that Alice would have sent in a $\mathbf{UR}^\subset$ or $\mathbf{UR}_k^\subset$ protocol, together with a random subset $B \subset S$ (using $\lceil \log_2 |B| \rceil + \lceil \log \binom{n}{|B|} \rceil$ bits, to represent both $|B|$ and the set B itself). Here $|B|$ is a random variable. These encoders are thus *Las Vegas*: the length of the encoding is a random variable, but the encoder/decoder always succeed in compressing and recovering the subset. The final lower bounds then come from the following simple lemma, which follows from the source coding theorem.

Lemma 2. *Let $\boldsymbol{s}$ denote the number of bits used by the $\mathbf{UR}^\subset$ or $\mathbf{UR}_k^\subset$ protocol, and let $\boldsymbol{s}'$ denote the expected number of bits to represent B. Then $(1 + \boldsymbol{s} + \boldsymbol{s}') \geq \log \binom{n}{m}$. In particular, $s \geq \log \binom{n}{m} - s' - 1$.*

Section III provides our first proof that $\mathbf{R}_\delta^{\rightarrow, pub}(\mathbf{UR}^\subset) = \Omega(\min\{n, \log^2(\frac{n}{\log(1/\delta)}) \log \frac{1}{\delta}\})$. We extend our results in the full version to $\mathbf{UR}_k^\subset$ for $k \geq 1$, proving a lower bound of $\Omega(k \log^2(n/k))$ communication even for constant failure probability.

B. Lower bound proof via reduction from $\mathbf{AugIndex}_N$

Our second lower bound proof for $\mathbf{UR}^\subset$ is via a randomized reduction from $\mathbf{AugIndex}_N$ [30]. In this problem, Charlie receives $z \in \{0, 1\}^N$ and Diane receives $j^* \in [N]$ together with z_j for $j = j^* + 1, \ldots, N$, and Diane must output z_{j^*}. It is shown in [30] that $\mathbf{R}_\delta^{\rightarrow, pub}(\mathbf{AugIndex}_N) = \Omega(N)$ for any δ bounded away from $1/2$. In our reduction, $N = \Theta(\log(1/\delta) \log^2 \frac{n}{\log(1/\delta)})$.

For $\mathbf{UR}^\subset$, we can also think of the problem as Alice being given $S \subseteq [n]$ and Bob being given $T \subsetneq S$, and Bob must output some element of $S \backslash T$. In $\mathbf{AugIndex}_N$, Charlie views his input as $L = \Theta(\log \frac{n}{\log(1/\delta)})$ blocks of bits of nearly equal size, where the ith block represents a subset S_i of $[u_i]$ in some collection $\mathcal{S}_{u_i, m}$ of sets, for some carefully chosen universe sizes u_i per block. Here $\mathcal{S}_{u_i, m}$ is a collection of subsets of $[u_i]$ of size m of maximal size such any two sets in the collection have intersection size strictly less than $m/2$. Furthermore, Diane's index j^* is in some particular block of bits corresponding to some set S_{i^*}, and Diane also knows S_i for $i > j$.

Now we explain the reduction. We assume some protocol $\mathcal{P}$ for $\mathbf{UR}^\subset$, and we give a protocol $\mathcal{P}'$ for $\mathbf{AugIndex}_N$. First, we define the universe $A = \bigcup_{i=1}^L (\{i\} \times [u_i] \times [100^i])$, which has size n. Charlie then defines $S = \bigcup_{i=1}^L (\{i\} \times S_i \times [100^i])$. Charlie and Diane use public randomness to define a uniformly random permutation π on $[n]$. Charlie can compute $\pi(S)$. Also, since Diane knows S_i for $i > i^*$, she can define $T = \bigcup_{i=i^*+1}^L (\{i\} \times S_i \times [100^i])$ and compute $\pi(T)$. Then $\pi(S)$ and $\pi(T)$ are the inputs to Alice and Bob in the protocol $\mathcal{P}$ for $\mathbf{UR}^\subset$. Charlie sends Diane the message that Alice would have sent Bob in $\mathcal{P}$ if her input had been $\pi(S)$, and Diane simulates Bob to recover an element in $\pi(S) \backslash \pi(T)$. Importantly, Alice and Bob do not know anything about π at this point other than that $\pi(S) = S$ and $\pi(T) = T$. Thus, the protocol $\mathcal{P}$ for $\mathbf{UR}^\subset$, if it succeeds, outputs an arbitrary element $j \in \pi(S) \backslash \pi(T)$, which is a deterministic function of the labels of elements in $\pi(S)$ and $\pi(T)$ and the randomness R that Alice and Bob share, which is independent from the randomness in π. Since π is still a uniformly random map conditioned on $\pi(S) = S$ and $\pi(t) = t$ for each $t \in T$, and $j \in \pi(S) \backslash \pi(T)$, it follows that $\pi^{-1}(t)$ is a uniformly random element of $S \backslash T$. After receiving $\pi^{-1}(j)$, if $(i, a, r) = \pi^{-1}(j)$, then Charlie and Diane reveal the pairs $((i, a, z), \pi((i, a, z)))$ for each $z \in [100^i]$ to Alice and Bob and Bob updates his set $\pi(T)$ to include $\pi(i, a, z)$ for each $z \in [100^i]$. One can show that at each step in this process, if Alice and Bob succeed in outputting an arbitrary item j from $\pi(S) \backslash \pi(T)$, then this is a uniformly random item from $\pi(S) \backslash \pi(T)$. The fact that this item is uniformly random is crucial for arguing the number of computation paths of the protocol of Alice and Bob is $o(1/\delta)$ with good probability, over π, so that one can argue (see below) that with good probability on every such computation path Alice and Bob succeed on that path, over their randomness R. Although the idea of using a random permutation appeared in [20] to show that any public coin $\mathbf{UR}$ protocol can be made into one in which a uniformly random element of $S \backslash T$ is output, here we must use this idea adaptively, slowly revealing

information about π and arguing that this property is maintained for each of Bob's successive queries.

Due to geometrically increasing repetitions of items for increasing i, a uniformly random element in $S \setminus T$ is roughly 100 times more likely to correspond to an item in S_{i^*} than in S_i for $i < i^*$. Thus if Diane simulates Bob to recover a random element in $S \setminus T$, it is most likely to recover an element j of S_{i^*}. She can then tell Bob to include $\pi(j)$ and its 100^{i^*} redundant copies to $\pi(T)$ and iterate.

There are several obstacles to overcome to make this work. First, iterating means using $\mathcal{P}$ adaptively, which was the same issue that arose in Section II-A. Second, a constant fraction of the time $(1/100)$, we expect to obtain an element *not* in S_{i^*}, but rather from some S_i for $i < i^*$. If this happened too often, then Diane would need to execute many queries to recover a sufficiently large number of elements from S_{i^*} in order to solve $\mathbf{AugIndex}_N$. This would then require a union bound over too many possible computation paths, which would not be possible as Alice likely would fail on one of them (over the choice of R). However, since the random permutation argument above ensures that at each step we receive a uniformly random item from the current set $S \setminus T$, if we continue for m iterations, we can argue that with large probability, our sequence of inputs T over the iterations with which Diane invokes Bob's output are all likely to come from a family $\mathcal{T}$ of size at most $2^{O(m)}$. Here we need to carefully construct this family to contain a smaller number of sets from levels i for which $i^* - i$ is larger so that the overall number of sets is small. Given this, we can union bound over all such T, for total failure probability $\delta |\mathcal{T}| \ll 1$. Furthermore, we can also argue that after m iterations, it is likely that we have recovered at least $m/2$ of the elements from S_{i^*}, which is enough to uniquely identify $S_{i^*} \in \mathcal{S}_{u_i, m}$ by the limited intersection property of $\mathcal{S}_{u_i, m}$.

III. Lower bounds via the adaptivity lemma

Consider a protocol $\mathcal{P}$ for $\mathbf{UR}^{\subset}$ with failure probability δ, operating in the one-way public coin model. When Alice's input is x and Bob's is y, Alice sends $\mathsf{Alice}(x)$ to Bob, and Bob outputs $\mathsf{Bob}(\mathsf{Alice}(x), y)$, which with probability at least $1 - \delta$ is in $support(x - y)$. As mentioned in Section II, we use $\mathcal{P}$ as a subroutine in a scheme for encoding/decoding elements of $\binom{[n]}{m}$ for $m = \lfloor \sqrt{n \log(1/\delta)} \rfloor$. We assume $\log \frac{1}{\delta} \leq n/64$, since for larger n we have an $\Omega(n)$ lower bound.

1) Encoding/decoding scheme: We now describe our encoding/decoding scheme $(\mathsf{ENC}, \mathsf{DEC})$ for elements in $\binom{[n]}{m}$, which uses $\mathcal{P}$ in a black-box way. The parameters shared by ENC and DEC are given in Algorithm 2.

As discussed in Section II, on input $S \in \binom{[n]}{m}$, ENC computes $M \leftarrow \mathsf{Alice}(\mathbf{1}_S)$ as part of its output. ENC also outputs a subset $B \subseteq S$ computed as follows. Initially $B = S$ and $S_0 = S$. ENC proceeds in R rounds. In round $r \in [R]$, ENC computes $s_r \leftarrow \mathsf{Bob}(M, \mathbf{1}_{S \setminus S_{r-1}})$. Let $b \in \{0, 1\}^R$ be such that b_r records whether Bob succeeds in round r. ENC also outputs b. If $s_r \in S_{r-1}$, i.e. $\mathsf{Bob}(M, \mathbf{1}_{S \setminus S_{r-1}})$ succeeds, ENC sets $b_r = 1$ and removes s_r from B (since the decoder can recover s_r from the $\mathbf{UR}^{\subset}$-protocol, ENC does not need to include it in B); otherwise ENC sets $b_r = 0$. At the end of round r, ENC picks a uniformly random set S_r in $\binom{S_{r-1} \setminus \{s_r\}}{n_r}$. In particular, ENC uses its shared randomness with DEC to generate S_r in such a way that $\mathsf{ENC}, \mathsf{DEC}$ agree on the sets S_r (DEC will actually iteratively construct $C_r = S \setminus S_r$). We present ENC in Algorithm 3.

The decoding process is symmetric. Let $C_0 = \emptyset$ and $A = \emptyset$. DEC proceeds in R rounds. On round $r \in [R]$, DEC obtains $s_r \in S \setminus C_{r-1}$ by invoking $\mathsf{Bob}(M, \mathbf{1}_{C_{r-1}})$. By construction of C_{r-1} (to be described later), it is guaranteed that $S_{r-1} = S \setminus C_{r-1}$. Therefore, DEC recovers exactly the same s_r as ENC. DEC initially assigns $C_r \leftarrow C_{r-1}$. If $b_r = 1$, DEC adds s_r to both A and C_r. At the end of round r, DEC inserts many random items from B into C_r so that $C_r = S \setminus S_r$. DEC can achieve this because of the shared random permutation π when constructing S_r. In the end, DEC outputs $B \cup A$. We present DEC in Algorithm 4.

Algorithm 2 Variables shared by encoder ENC and decoder DEC.

1: $m \leftarrow \lfloor \sqrt{n \log \frac{1}{\delta}} \rfloor$
2: $K \leftarrow \lfloor \frac{1}{16} \log \frac{1}{\delta} \rfloor$
3: $R \leftarrow \lfloor K \log(m/4K) \rfloor$
4: **for** $r = 0, \dots, R$ **do**
5: $\quad n_r \leftarrow \lfloor m \cdot 2^{-\frac{r}{K}} \rfloor$
$\quad\quad\quad\quad \triangleright |S_r| = n_r$, and $\forall r\ n_r - n_{r+1} \geq 2$
6: **end for**
7: π is a random permutation on $[n]$
$\quad\quad\quad\quad\quad\quad\quad \triangleright$ Used to generate S_r and C_r

---

2) Analysis: We have two random objects in our encoding/decoding scheme: (1) the random source used by $\mathcal{P}$, denoted by X, and (2) the random permutation π. These are independent.

First, we can prove that $\mathsf{DEC}(\mathsf{ENC}(S)) = S$. That is, for any fixing of the randomness in X and π, DEC will always decode S successfully. It is because ENC and DEC share X and π, so that DEC essentially simulates ENC. We formally prove this by induction in Lemma 3.

Now our goal is to prove that by using the $\mathbf{UR}^{\subset}$-protocol, the number of bits that ENC saves in expectation over the naive $\lceil \log\binom{n}{m} \rceil$-bit encoding is

Algorithm 3 Encoder ENC.

1: **procedure** ENC(S)
2:     $M \leftarrow \mathsf{Alice}(\mathbf{1}_S)$
3:     $A \leftarrow \emptyset$ ▷ the set DEC recovers just from M
4:     $S_0 \leftarrow S$
      ▷ at end of round r, DEC still needs to recover S_r
5:     **for** $r = 1, \ldots, R$ **do**
6:         $s_r \leftarrow \mathsf{Bob}(M, \mathbf{1}_{S \setminus S_{r-1}})$
                    ▷ $s_r \overset{?}{\in} S_{r-1}$ found in round r
7:         $S_r \leftarrow S_{r-1}$
8:         **if** $s_r \in S_{r-1}$ **then**
                        ▷ i.e. if s_r is a valid sample
9:             $b_r \leftarrow 1$
          ▷ $b \in \{0,1\}^R$ indicating which rounds succeed
10:             $A \leftarrow A \cup \{s_r\}$
11:             $S_r \leftarrow S_r \setminus \{s_r\}$
12:         **else**
13:             $b_r \leftarrow 0$
14:         **end if**
15:         Remove $|S_r| - n_r$ elements from S_r with
        smallest π_a's among $a \in S_r$ ▷ now $|S_r| = n_r$
16:     **end for**
17:     **return** (M, $S \setminus A$, b)
18: **end procedure**

Algorithm 4 Decoder DEC.

1: **procedure** DEC(M, B, b)
                    ▷ M is $\mathsf{Alice}(\mathbf{1}_S)$
    ▷ $b \in \{0,1\}^R$ indicates rounds in which Bob succeeds
    ▷ B contains all elements of S that DEC doesn't recover via M
2:     $A \leftarrow \emptyset$
      ▷ the subset of S DEC recovers just from M
3:     $C_0 \leftarrow \emptyset$ ▷ subset of S we have built up so far
4:     **for** $r = 1, \ldots, R$ **do**
    ▷ each iteration tries to recover 1 element via M
5:         $C_r \leftarrow C_{r-1}$
6:         **if** $b_r = 1$ **then**
                ▷ this means Bob succeeds in round r
7:             $s_r \leftarrow \mathsf{Bob}(M, \mathbf{1}_{C_{r-1}})$
        ▷ Invariant: $C_r = S \setminus S_r$ (S_r is defined in ENC)
8:             $A \leftarrow A \cup \{s_r\}$
9:             $C_r \leftarrow C_r \cup \{s_r\}$
10:         **end if**
11:         Insert $m - n_r - |C_r|$ items into C_r with
        smallest π_a's among $a \in B \setminus C_r$
        ▷ Random masking "Differential Privacy" step.
        Still n_r elements left to recover.
12:     **end for**
13:     **return** $B \cup A$
14: **end procedure**

$\Omega(\log \frac{1}{\delta} \log^2 \frac{n}{\log(1/\delta)})$ bits. Intuitively, it is equivalent to prove the number of elements that ENC saves is $\Omega(\log \frac{1}{\delta} \log \frac{n}{\log(1/\delta)})$. We formalize this in Lemma 4. Note that ENC also needs to output b (i.e., whether the Bob succeeds on R rounds), which takes R bits. By our setting of parameters, we can afford the loss of R bits. Thus it is sufficient to prove $\mathbb{E}|B| = |S| - \Omega(\log \frac{1}{\delta} \log \frac{n}{\log(1/\delta)})$.

We have $|S| - |B| = \sum_{r=1}^{R} b_r$. In Lemma 1, we prove the probability that Bob fails on round r is upper bounded by $\frac{I(X; S_{r-1}) + 1}{\log \frac{1}{\delta}}$, where $I(X; S_{r-1})$ is the mutual information between X and S_{r-1}. Furthermore, we will show in Lemma 5 that $I(X; S_{r-1})$ is upper bounded by $O(K)$. By our setting of parameters, we have $\mathbb{E} b_r = \Omega(1)$ and thus $\mathbb{E}(|S| - |B|) = \Omega(R) = \Omega(\log \frac{1}{\delta} \log \frac{n}{\log(1/\delta)})$.

Lemma 3. *DEC*(*ENC*(S)) = S.

Proof: We claim that for $r = 0, \ldots, R$, $\{S_r, C_r\}$ is a partition of S (S_r is defined in Algorithm 3, and C_r in Algorithm 4). We prove the claim by induction on r. Our base case is $r = 0$, for which the claim holds since $S_0 = S$, $C_0 = \emptyset$.

Assume the claim holds for $r - 1$ ($1 \le r \le R$), and we consider round r. On round r, by induction

$S \setminus S_{r-1} = C_{r-1}$, the index s_r obtained by both ENC and DEC are the same. Initially $S_r = S_{r-1}$ and $C_r = C_{r-1}$, and so $\{S_r, C_r\}$ is a partition of S. If s_r is a valid sample (i.e. $s_r \in S_{r-1}$), then $b_r = 1$, and ENC removes s_r from S_r and in the meanwhile DEC inserts s_r into C_r, so that $\{S_r, C_r\}$ remains a partition of S. Next, ENC repeats removing the a from S_r with the smallest π_a value until $|S_r| = n_r$. Symmetrically, DEC repeats inserting the a into C_r with the smallest π_a value among $a \in B \setminus C_r$, until $|C_r| = |S| - n_r$. In the end we have $|S_r| + |C_r| = |S|$, so ENC and DEC execute repetition the same number of times. Moreover, we can prove that during the same iteration of this repeated insertion, the element removed from S_r is exactly the same element inserted to C_r. This is because in the beginning of a repetition $\{S_r, C_r\}$ is a partition of S. We have $B \setminus C_r \subseteq S \setminus C_r = S_r$. Let a^* denote $a \in S_r$ that minimizes π_a. Then $a^* \in B \setminus C_r \subseteq S_r$ (since a^* will be removed from S_r, it has no chance to be included in S in ENC, so that B contains a^*), and π_{a^*} is also the smallest among $\{\pi_a : a \in B \setminus C_r\}$. Thus both ENC and DEC will take a^* (for ENC, to remove from S_r, and for DEC, to insert into C_r). Therefore, $\{S_r, C_r\}$ remains a partition of S.

Given the fact that $\{S_r, C_r\}$ is a partition of S, the s_r are the same in ENC and DEC. Furthermore, $A = \{s_r : b_r = 1, r = 1, \ldots, R\}$ are the same in ENC and DEC. We know $A \subseteq S$. Since ENC outputs $S \backslash A$, and DEC outputs $(S \backslash A) \cup A$, we have $\mathsf{DEC}(\mathsf{ENC}(S)) = S$. ∎

The proof of the following lemma appears in the full version.

Lemma 4. *Let $W \in \mathbb{N}$ be a random variable with $W \leq m$ and $\mathbb{E} W \leq m - d$. Then $\mathbb{E}(\log \binom{n}{m} - \log \binom{n}{W}) \geq d \log(\frac{n}{m} - 1)$.*

Lemma 1 (restated). Consider $f: \{0,1\}^b \times \{0,1\}^q \to \{0,1\}$ and $X \in \{0,1\}^b$ uniformly random. If $\forall y \in \{0,1\}^q$, $\mathbb{P}(f(X,y) = 1) \leq \delta$ where $0 < \delta < 1$, then for any r.v. Y supported on $\{0,1\}^q$,

$$\mathbb{P}(f(X,Y) = 1) \leq \frac{I(X;Y) + H_2(\delta)}{\log \frac{1}{\delta}},$$

where $I(X;Y)$ is the mutual information between X and Y, and H_2 is the binary entropy function.

Proof: It is equivalent to prove $I(X;Y) \geq \mathbb{E}(f(X,Y)) \cdot \log \frac{1}{\delta} - H_2(\delta)$. By definition of mutual entropy $I(X;Y) = H(X) - H(X|Y)$, where $H(X) = b$ and we must show $H(X|Y) \leq H_2(\delta) + (1 - \mathbb{E}(f(X,Y))) \cdot b + \mathbb{E}(f(X,Y)) \cdot (b - \log \frac{1}{\delta}) = b + H_2(\delta) - \mathbb{E}(f(X,Y)) \cdot \log \frac{1}{\delta}$.

The upper bound for $H(X|Y)$ is obtained by considering the following one-way communication problem: Alice knows both X and Y while Bob only knows Y, and Alice must send a single message to Bob so that Bob can recover X. The expected message length in an optimal protocol is exactly $H(X|Y)$. Thus, any protocol gives an upper bound for $H(X|Y)$, and we simply take the following protocol: Alice prepends a 1 bit to her message iff $f(X,Y) = 1$ (taking $H_2(\delta)$ bits in expectation). Then if $f(X,Y) = 0$, Alice sends X directly (taking b bits). Otherwise, when $f(X,Y) = 1$, Alice sends the index of X in $\{x | f(x,Y) = 1\}$ (taking $\log(\delta 2^b) = b - \log \frac{1}{\delta}$ bits). ∎

Corollary 1. *Let X denote the random source used by the $\mathbf{UR}^\subset$-protocol with failure probability at most δ. If S is a fixed set and $T \subset S$, $\mathbb{P}(\mathsf{Bob}(\mathsf{Alice}(\mathbf{1}_S), \mathbf{1}_T) \notin S \backslash T) \leq \frac{I(X;T) + H_2(\delta)}{\log \frac{1}{\delta}}$.*

Lemma 5. *$I(X; S_r) \leq 6K$, for $r = 1, \ldots, R$.*

Proof: Note that $I(X; S_r) = H(S_r) - H(S_r|X)$. Since $|S_r| = n_r$ and $S_r \subseteq S$, $H(S_r) \leq \log \binom{m}{n_r}$. Here is the main idea to lower bound $H(S_r|X)$: By definition of conditional entropy, $H(S_r|X) = \sum_x p_x \cdot H(S_r|X = x)$. We fix an arbitrary x. If we can prove that for any $T \subseteq S$ where $|T| = n_r$,

$\mathbb{P}(S_r = T|X = x) \leq p$, then by definition of entropy we have $H(S_r|X = x) \geq \log \frac{1}{p}$. This is shown in the full version for $p = 2^{6K}/\binom{m}{n_r}$. ∎

Theorem 2. $\mathbf{R}_\delta^{\to, pub}(\mathbf{UR}^\subset) = \Omega(\log \frac{1}{\delta} \log^2 \frac{n}{\log(1/\delta)})$, *given that $64 \leq \log \frac{1}{\delta} \leq \frac{n}{64}$.*

Proof: By Lemma 3, the success probability of protocol $(\mathsf{ENC}, \mathsf{DEC})$ is 1. By Lemma 2, we have $\mathsf{s} \geq \log \binom{n}{m} - \mathsf{s}' - 1$, where $\mathsf{s}' = \log n + R + \mathbb{E}(\log \binom{n}{|B|}))$. The size of B is $|B| = |S| - \sum_{r=1}^R b_r$. By Corollary 1, conditioned on S, $\mathbb{P}(b_r = 0) \leq \frac{I(X; S_{r-1}) + 1}{\log \frac{1}{\delta}}$. By Lemma 5, $I(X; S_{r-1}) \leq 6K$ (Note that when $r = 1$, $I(X; S_0) = 0 \leq 6K$). Therefore, $\mathbb{E}(b_r) \geq 1 - \frac{6K+1}{\log \frac{1}{\delta}}$. By the setting of parameters (see Algorithm 2) we have $\mathbb{E}(b_r) \geq \frac{39}{64}$. Therefore, $\mathbb{E}(|B|) \leq |S| - \frac{39}{64} R$. By Lemma 4, $\log \binom{n}{m} - \mathbb{E}(\log \binom{n}{|B|})) \geq \frac{39}{64} R \cdot \log(\frac{n}{m} - 1) \geq \frac{1}{2} R \log(\frac{n}{\log(1/\delta)})$. Furthermore, $\frac{1}{6} R \log \frac{n}{\log(1/\delta)} \geq R$. Thus we obtain $\mathsf{s} \geq \frac{R}{3} \log \frac{n}{\log(1/\delta)} - (\log n + 1) = \Omega(\log \frac{1}{\delta} \log^2 \frac{n}{\log(1/\delta)})$. ∎

IV. LOWER BOUNDS PROOFS VIA AUGMENTED INDEXING

Here we show another route to proving $\mathbf{R}_\delta^{\to, pub}(\mathbf{UR}_k^\subset) = \Omega(\min\{n, t \log^2(n/t)\})$ via reduction from augmented indexing. We again separately prove lower bounds for $\mathbf{R}_\delta^{\to, pub}(\mathbf{UR}^\subset)$ and $\mathbf{R}_{\frac{1}{5}}^{\to, pub}(\mathbf{UR}_k^\subset)$. Both proofs make use of the following standard lemma. The proof can be found in the full version [23].

Lemma 6. *For any integers $u \geq 1$ and $1 \leq m \leq u/(4e)$, there exists a collection $\mathcal{S}_{u,m} \subset \binom{[u]}{m}$ with $\log |\mathcal{S}_{u,m}| = \Theta(m \log(u/m))$ such that for all $S \neq S' \in \mathcal{S}_{u,m}$, $|S \cap S'| < m/2$.*

Both our lower bounds (for $\mathbf{UR}^\subset$ and $\mathbf{UR}_k^\subset$) reduce from augmented indexing (henceforth **AugIndex**) to either $\mathbf{UR}^\subset$ with low failure probability, or $\mathbf{UR}_k^\subset$ with constant failure probability, in the public coin one-way model of communication. We remind the reader of the setup for the $\mathbf{AugIndex}_N$ problem. There are two players, Charlie and Diane. Charlie receives $z \in \{0,1\}^N$ and Diane receives $j^* \in [N]$ together with $z_{j^*+1}, \ldots, z_N$. Charlie must send a single message to Diane such that Diane can then output z_{j^*}. The following theorem is known.

Theorem 3. *[30] $\mathbf{R}_{1/3}^{\to, pub}(\mathbf{AugIndex}_N) = \Theta(N)$.*

We show that if there is an s-bit communication protocol $\mathcal{P}$ for $\mathbf{UR}^\subset$ on n-bit vectors with failure probability δ (or for $\mathbf{UR}_k$ with constant failure probability), that implies the existence of an s-bit protocol $\mathcal{P}'$ for

AugIndex$_N$ for some $N = \Theta(\log \frac{1}{\delta} \log^2 \frac{n}{\log \frac{1}{\delta}})$ (or $N = \Theta(k \log^2(n/k))$ for $\mathbf{UR}_k$). The lower bound on s then follows from Theorem 3.

Set $t = \log \frac{1}{\delta}$. In this section we assume $t < n/(4e)$ and show $\mathbf{R}_\delta^{\rightarrow,pub}(\mathbf{UR}^\subset) = \Omega(t \log^2(n/t))$. This implies a lower bound of $\Omega(\min\{n, t \log^2(n/t)\})$ for all $\delta > 0$ bounded away from 1.

As mentioned, we assume we have an s-bit protocol $\mathcal{P}$ for $\mathbf{UR}^\subset$ with failure probability δ, with players Alice and Bob. We use $\mathcal{P}$ to give an s-bit protocol $\mathcal{P}'$ for **AugIndex**$_N$, which has players Charlie and Diane, for $N = \Theta(t \log^2(n/t))$.

Algorithm 5 Behavior of Diane in $\mathcal{P}'$ for $\mathbf{UR}^\subset$.

1: **procedure** Diane(M)
2: $\quad T \leftarrow \bigcup_{i=i^*+1}^{L}(\{i\} \times S_i \times [100^i])$
3: $\quad T_{i^*} \leftarrow \emptyset$
4: $\quad$ **while** $|T_{i^*}| < \frac{m}{2}$ **do**
5: $\qquad (i, a, r) \leftarrow \pi^{-1}(\mathsf{Bob}(M, \mathbf{1}_{\pi(T)}))$
6: $\qquad T \leftarrow T \cup ((i, a) \times [100^i])$
7: $\qquad$ **if** $i = i^*$ **then**
8: $\qquad\quad T_{i^*} \leftarrow T_{i^*} \cup \{a\}$
9: $\qquad$ **end if**
10: $\quad$ **end while**
11: $\quad$ **if** there exists $S \in \mathcal{S}_{u_{i^*},m}$ with $T_{i^*} \subset S$ **then**
12: $\qquad$ **return** the unique such S
13: $\quad$ **else**
14: $\qquad$ **return** Fail
15: $\quad$ **end if**
16: **end procedure**

The protocol $\mathcal{P}'$ operates as follows. Without loss of generality we may assume that, using the notation of Lemma 6, $|\mathcal{S}_{u,m}|$ is a power of 2 for u, m as in the lemma statement. This is accomplished by simply rounding $|\mathcal{S}_{u,m}|$ down to the nearest power of 2 by removing elements arbitrarily. Also, define $L = c \log(n/t)$ for some sufficiently small constant $c \in (0, 1)$ to be determined later. Now, Charlie partitions the bits of his input $z \in \{0, 1\}^N$ into L consecutive sequences of bits such that the ith chunk of bits for each $i \in [L]$ can be viewed as specifying an element $S_i \in \mathcal{S}_{u_i,m}$ for $u_i = \frac{n}{100^i \cdot L}$ and $m = ct$. Lemma 6 gives $\log |\mathcal{S}_{u_i,m}| = \Theta(m \log(u_i/m))$, which is $\Theta(t \log(n/t))$ for $c < 1/14$. Thus $N = \Theta(L \cdot t \log(n/t)) = \Theta(t \log^2(n/t))$. Given these sets $S_1, \dots, S_L$, we now discuss how Charlie generates a vector $x \in \{0, 1\}^n$. Charlie then simulates Alice on x to generate the message Alice would have send to Bob in protocol $\mathcal{P}$, then sends that same message to Diane.

To generate $x \in \{0, 1\}^n$, assume Charlie and Diane have sampled a bijection from $A = \bigcup_{i=1}^{L}(\{i\} \times [u_i] \times$

$[100^i])$ to $[n]$ uniformly at random. We denote this bijection by π. This is possible since $|A| = n$. Then Charlie defines x to be the indicator vector $\mathbf{1}_{\pi(S)}$, where $S = \bigcup_{i=1}^{L}(\{i\} \times S_i \times [100^i])$, then sends a message M to Diane, equal to Alice's message with input $\mathbf{1}_{\pi(S)}$. This completes the description of Charlie's behavior in the protocol $\mathcal{P}'$.

We describe how Diane uses M to solve **AugIndex**$_N$. Diane's input $j^* \in [N]$ lies in some chunk $i^* \in [L]$. We now show how Diane can use $\mathcal{P}$ to recover S_{i^*} with probability $2/3$ (and thus in particular recover z_{j^*}). Since Diane knows z_j for $j > j^*$, she knows S_i for $i > i^*$. She can then execute the following algorithm.

In Algorithm 5 Diane is building up a subset T_{i^*} of S_{i^*}. Once $|T_{i^*}| \geq |S_{i^*}|/2 = m/2$, Diane can uniquely recover S_{i^*} by the limited intersection property of $\mathcal{S}_{u_i,m}$ guaranteed by Lemma 6. Until then, she uses $\mathcal{P}$ to recover elements of $S \setminus T$, which, as we now show, are chosen uniformly at random from $S \setminus T$.

The proof of the following claim is deferred to the full version.

Claim 4. *For every protocol for Alice and Bob that uses shared randomness with Bob's behaviour given by* $\mathsf{Bob}(\cdot)$, *for every choice of shared random string R of Alice and Bob, for every $S, T \subset S$, the following conditions hold. If π is a uniformly random permutation, the success or failure of $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)})$ is determined by $\{\pi(j)\}_{j \in T}$ and the image $\pi(S \setminus T)$ of $S \setminus T$ under π. Conditioned on a choice of R, $\{\pi(j)\}_{j \in T}$ and $\pi(S \setminus T)$ such that $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)})$ succeeds, one has that $\pi^{-1}(\mathsf{Bob}(M, \mathbf{1}_{\pi(T)}))$ is a uniformly random element of $S \setminus T$.*

Fix any protocol $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ (not necessarily the one that Charlie and Diane use; see analysis of the idealized process $\widetilde{\mathcal{P}}$ below). Now fix T together with values of R, $\{\pi(j)\}_{j \in T}$ and $\pi(S \setminus T)$ such that $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ succeeds.

Elements in $S_j, j < i^*$, **are unlikely to be recovered.** Given Claim 4, since the elements of S_j appear with frequency 100^j in $S \setminus T$, they are less likely to be returned by $\pi^{-1}(\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)}))$ for small j. Specifically, as long as $|S_{i^*} \cap T_{i^*}| \leq m/2$, for any $j < i^*$ $\mathbb{P}(i = j | (R, \{\pi(j)\}_{j \in T}, \pi(S \setminus T))$ s.t. $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ succeeds$) \leq \frac{m \cdot 100^j}{\frac{m}{2} \cdot 100^{i^*}} \leq 2 \cdot 100^{-(i^*-j)} \leq 50^{-(i^*-j)}$. Here again the probability is over the choice of $\pi|_{S \setminus T} : (S \setminus T) \to \pi(S \setminus T)$ (recall that we condition on the image $\pi(S \setminus T)$ under π, but not on the actual mapping).

We now define the set $\mathcal{T}$ of **typical intermediate sets**, which plays a crucial role in our analysis. Let Q_i

for $i \in [L]$ denote $\{i\} \times S_i \times [100^i]$. Let $\mathcal{T}$ be the collection of all $T \subset S$ such that (1) $Q_i \subset T$ for all $i > i^*$, and (2) for each $i < i^*$, $|T \cap Q_i| \leq 100^i \cdot m/4^{i^*-i}$, such that if T contains (i, a, b) for some $a \in S_i$ and $b \in [100^i]$, then T contains $(i, a) \times [100^i]$. The following claim is proven in the full version:

Claim 5. *The set $\mathcal{T}$ as above satisifies $|\mathcal{T}| = 2^{O(m)}$.*

We will show that for most choices of π and shared random string R Algorithm 5 **(a)** never leaves the set $\mathcal{T}$ and **(b)** successfully terminates. Note that Algorithm 5 is a random process whose sample space is the product of the set of all possible permutations π and shared random strings R. As before, we denote this process by $\mathcal{P}'$. It is useful for analysis purposes to define another process $\widetilde{\mathcal{P}}$, which is an idealized version of $\mathcal{P}'$. In this process instead of running $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)})$ Alice runs $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$, which is guaranteed to output an element of $\pi(S \setminus T)$ for every choice of $T \subset S$, shared random string R, $\{\pi(j)\}_{j \in T}$, and $\pi(S \setminus T)$. The proof proceeds in three steps.

Step 1: proving that $\widetilde{\mathcal{P}}$ succeeds in recovering T_{i^*} and never leaves $\mathcal{T}$ with high probability. Choose π uniformly at random. By the upper bound on returning an element of S_i above, as long as $|S_{i^*} \cap T_{i^*}| \leq m/2$, the expected number of items recovered by Bob from S_i for $i < i^*$ in the first m iterations is at most $m/50^{i^*-i}$. Thus the probability of recovering more than $m/4^{i^*-i}$ items from S_i is at most $(1/12)^{i^*-i}$ by Markov's inequality. Note that the probability is over the choice of π only, as $\widetilde{\mathsf{Bob}}$ is assumed to succeed with probability 1 by definition of $\widetilde{\mathcal{P}}$. Thus $\mathbb{P}(\widetilde{\mathcal{P}}$ leaves $\mathcal{T}) \leq \sum_{i=1}^{i^*-1} (1/12)^{i^*-i} < 1/10$. In particular this means that with probability at least $1 - 1/10$ at most $\sum_{i < i^*} m/4^{i^*-i} < m/2$ items from $\bigcup_{i < i^*} S_i$ are recovered in the first m (or fewer, if the algorithm terminates earlier) iterations. This also implies that with probability at least $1 - 1/10$ if the algorithm proceeds for the entire m iterations, it recovers at least $m/2$ elements of T_{i^*} and hence terminates. We thus get that $\widetilde{\mathcal{P}}$ succeeds at least with probability $1 - 1/10$.

Step 2: coupling $\widetilde{\mathcal{P}}$ to $\mathcal{P}'$ on most of the probability space. For every $T \subset S$ and every π let $\mathcal{E}_T(\pi)$ be the probabilistic event (over the choice of Bob's random string R) that $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)})$ succeeds in returning an element in $\pi(S \setminus T)$. Note that $\mathcal{E}_T(\pi)$ is a subset of the probability space of shared random strings R, and depends on π. We let $\mathcal{E}_{\mathcal{T}}(\pi) := \wedge_{T \in \mathcal{T}} \mathcal{E}_T(\pi)$ to simplify notation. Using Claim 5 and the union bound we have for every π $\mathbb{P}_R(\neg(\mathcal{E}_{\mathcal{T}}(\pi))) \leq \delta \cdot |\mathcal{T}| \leq 1/20$ as long as for $m = c \log(1/\delta)$ for c a sufficiently small constant.

Now recall that $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ is an idealized protocol, which is guaranteed to output an element of $\pi(S \setminus T)$ for every choice of $T \subset S$, shared random string R, $\{\pi(j)\}_{j \in T}$, and $\pi(S \setminus T)$. We have just shown that for every π the event $\mathcal{E}_T(\pi)$ occurs with probability at least $1 - 1/20$ over the choice of R. Now define $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ as equal to $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)})$ for all $T \in \mathcal{T}$ (the typical set of intermediate sets) and (π, R) such that $R \in \mathcal{E}_T(\pi)$, and extend $\widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ to return an arbitrary element of $\pi(S \setminus T)$ for remaining tuples $(T, R, \pi(T), \pi(S \setminus T))$. Note that $\widetilde{\mathsf{Bob}}$ defined in this way is a deterministic function once T, R, $\pi(T)$ and $\pi(S \setminus T)$ are fixed. Note that with probability at least $1 - 1/20$ over the choice of π and R one has $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)}) = \widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ for all $T \in \mathcal{T}$, as required.

Step 3: arguing that $\mathcal{P}'$ succeeds with high probability. Choose (π, R) uniformly at random. By **Step 2** we have that with probability at least $1 - 1/20$ over this choice $\mathsf{Bob}(M, \mathbf{1}_{\pi(T)}) = \widetilde{\mathsf{Bob}}(M, \mathbf{1}_{\pi(T)})$ for all $T \in \mathcal{T}$. At the same time we have by **Step 1** that with probability at least $1 - 1/10$ over the choice of π the idealized process $\widetilde{\mathcal{P}}$ succeeds in recovering T_{i^*} and never leaves $\mathcal{T}$. Putting the two bounds together, we get that $\mathcal{P}'$ succeeds with probability at least $1 - 1/20 - 1/10 > 2/3$, showing the following theorem.

Theorem 4. *For any $0 < \delta < 1/2$ and integer $n \geq 1$ with $\log \frac{1}{\delta} < n/(4e)$, $\mathbf{R}_{\delta}^{\rightarrow, pub}(\mathbf{UR}^{\subset}) \geq \mathbf{R}_{1/3}^{\rightarrow, pub}(\mathbf{AugIndex}_N)$ for $N = \Theta(\log \frac{1}{\delta} \log^2 \frac{n}{\log \frac{1}{\delta}})$.*

Corollary 2. *For any $0 < \delta < 1/2$ and integer $n \geq 1$, $\mathbf{R}_{\delta}^{\rightarrow, pub}(\mathbf{UR}^{\subset}) = \Omega(\min\{n, \log \frac{1}{\delta} \log^2 \frac{n}{\log \frac{1}{\delta}}\})$.*

In the full version [23] we use similar, but slightly simpler, ideas to lower bound $\mathbf{R}_{\frac{1}{5}}^{\rightarrow, pub}(\mathbf{UR}_k^{\subset})$.

References

[1] K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 459–467, 2012.

[2] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 5–14, 2012.

[3] K. J. Ahn, S. Guha, and A. McGregor. Spectral sparsification in dynamic graph streams. In *Proceedings of the 16th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 1–10, 2013.

[4] A. Andoni, R. Krauthgamer, and K. Onak. Streaming algorithms via precision sampling. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 363–372, 2011.

[5] S. Assadi, S. Khanna, and Y. Li. On estimating maximum matching size in graph streams. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1723–1742, 2017.

[6] S. Assadi, S. Khanna, Y. Li, and G. Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1345–1364, 2016.

[7] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing (STOC)*, pages 173–182, 2015.

[8] M. Bury and C. Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*, pages 263–274, 2015.

[9] R. Chitnis, G. Cormode, H. Esfandiari, M. Hajiaghayi, A. Mc-Gregor, M. Monemizadeh, and S. Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1326–1344, 2016.

[10] R. H. Chitnis, G. Cormode, M. T. Hajiaghayi, and M. Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1234–1251, 2015.

[11] D. Coppersmith and R. Kumar. An improved data stream algorithm for frequency moments. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 151–156, 2004.

[12] G. Cormode, S. Muthukrishnan, and I. Rozenbaum. Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 25–36, 2005.

[13] H. Esfandiari, M. Hajiaghayi, and D. P. Woodruff. Brief announcement: Applications of uniform sampling: Densest subgraph and beyond. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 397–399, 2016.

[14] M. Farach-Colton and M. Tsai. Tight approximations of degeneracy in large graphs. In *Proceedings of the 12th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 429–440, 2016.

[15] G. Frahling, P. Indyk, and C. Sohler. Sampling in dynamic data streams and applications. *Int. J. Comput. Geometry Appl.*, 18(1/2):3–28, 2008. Preliminary version in SOCG 2005.

[16] D. Gibb, B. M. Kapron, V. King, and N. Thorn. Dynamic graph connectivity with improved worst case update time and sublinear space. *CoRR*, abs/1509.06464, 2015.

[17] P. Gopalan and J. Radhakrishnan. Finding duplicates in a data stream. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 402–411, 2009.

[18] S. Guha, A. McGregor, and D. Tench. Vertex and hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems (PODS)*, pages 241–247, 2015.

[19] J. W. Hegeman, G. Pandurangan, S. V. Pemmaraju, V. B. Sardeshmukh, and M. Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and MST. In *Proceedings of the 34th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 91–100, 2015.

[20] H. Jowhari, M. Sağlam, and G. Tardos. Tight bounds for Lp samplers, finding duplicates in streams, and related problems.

In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 49–58. ACM, 2011.

[21] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 41–52, 2010.

[22] M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford. Single pass spectral sparsification in dynamic streams. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 561–570, 2014.

[23] M. Kapralov, J. Nelson, J. Pachocki, Z. Wang, D. P. Woodruff, and M. Yahyazadeh. Optimal lower bounds for universal relation, and for samplers and finding duplicates in streams. *CoRR*, abs/1704.00633, 2017.

[24] B. M. Kapron, V. King, and B. Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1131–1142, 2013.

[25] M. Karchmer, R. Raz, and A. Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3-4):191–204, 1995.

[26] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. Discrete Math.*, 3(2):255–265, 1990.

[27] C. Konrad. Maximum matching in turnstile streams. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*, pages 840–852, 2015.

[28] A. McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014.

[29] A. McGregor, D. Tench, S. Vorotnikova, and H. T. Vu. Densest subgraph in dynamic graph streams. In *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 472–482, 2015.

[30] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.*, 57(1):37–49, 1998.

[31] M. Monemizadeh and D. P. Woodruff. 1-pass relative-error l_p-sampling with applications. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1143–1160, 2010.

[32] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1(2):117–236, 2005.

[33] J. Nelson, J. Pachocki, and Z. Wang. Optimal lower bounds for universal relation, samplers, and finding duplicates. *CoRR*, abs/1703.08139, Mar. 2017.

[34] G. Pandurangan, P. Robinson, and M. Scquizzato. Fast distributed algorithms for connectivity and MST in large graphs. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 429–438, 2016.

[35] G. Tardos and U. Zwick. The communication complexity of the universal relation. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity (CCC)*, pages 247–259, 1997.

[36] J. Tarui. Finding a duplicate and a missing item in a stream. In *Proceedings of the 4th Annual Conference on Theory and Applications of Models of Computation (TAMC)*, pages 128–135, 2007.

[37] Z. Wang. An improved randomized data structure for dynamic graph connectivity. *CoRR*, abs/1510.04590, 2015.